Traffictelligence – Advanced Traffic Volume Estimation with Machine Learning

# 1. Introduction

Project Title: Traffictelligence – Advanced Traffic Volume Estimation with Machine Learning

Team ID: LTVIP2025TMID32613  
Team Leader: B Venkata Mohan Reddy  
Team Member: Aienampudi Suhnitha

# 2. Project Overview

Purpose:  
Traffictelligence is a machine learning-powered web application designed to estimate traffic volume based on user-input features such as weather conditions, date, time, and holiday indicators. It aids in understanding traffic patterns for better decision-making by urban planners, authorities, and researchers.

Key Features:  
- Clean and responsive UI built with HTML and CSS  
- Dropdown-based form for manual traffic feature input  
- Flask backend integrated with a trained ML model  
- Real-time prediction of traffic volume  
- Server-side rendering with Jinja2  
- Handles edge cases and invalid inputs gracefully

# 3. Architecture

Frontend:  
- HTML5, CSS3  
- Jinja2 templating via Flask  
  
Backend:  
- Flask (Python microframework)  
- Trained ML model loaded using joblib or safetensors  
- Form submission through POST method and result rendering  
  
No database is required, as this is a lightweight prediction tool.

# 4. Setup Instructions

Prerequisites:  
- Python 3.x (recommended: 3.10 or lower for compatibility)  
- pip (Python package installer)  
  
Installation Steps:  
1. Clone the repository:  
 git clone https://github.com/Suhnitha/Traffic-Intelligence-App.git  
  
2. Navigate to the project folder:  
 cd Traffic-Intelligence-App  
  
3. Create a virtual environment (optional but recommended):  
 python -m venv venv  
 source venv/bin/activate # On Windows: venv\Scripts\activate  
  
4. Install required Python packages:  
 pip install -r requirements.txt  
  
5. Launch the Flask app:  
 python app.py

# 5. Folder Structure

traffictelligence-flask/  
│  
├── templates/ # HTML templates (Jinja2)  
│ └── index.html   
├── app.py # Flask application logic  
├── model.pkl/ # Trained model

├── requirements.txt # Python dependencies  
└── README.md # Project documentation

# 6. Running the Application

To start the app:  
  
python app.py  
  
Then open your browser at:  
http://127.0.0.1:5000/  
  
Fill the form with required traffic-related details and click “Predict” to get the estimated traffic volume.

# 7. Model Information

- Trained on historical traffic datasets  
- Features include: holiday, weather, temperature, date & time, rain, snow  
- Model: Can be a regression model like RandomForest, or a fine-tuned T5 transformer for categorical-to-text prediction

# 8. User Interface

- Simple and mobile-friendly design  
- Form-based input for easy usability  
- Error handling for invalid or incomplete input  
- Displays predicted traffic volume clearly

# 9. Testing

- Manual testing through form submission  
- Edge case checks for out-of-range inputs  
- Backend prediction logic tested using dummy inputs

# 10. Known Issues / Limitations

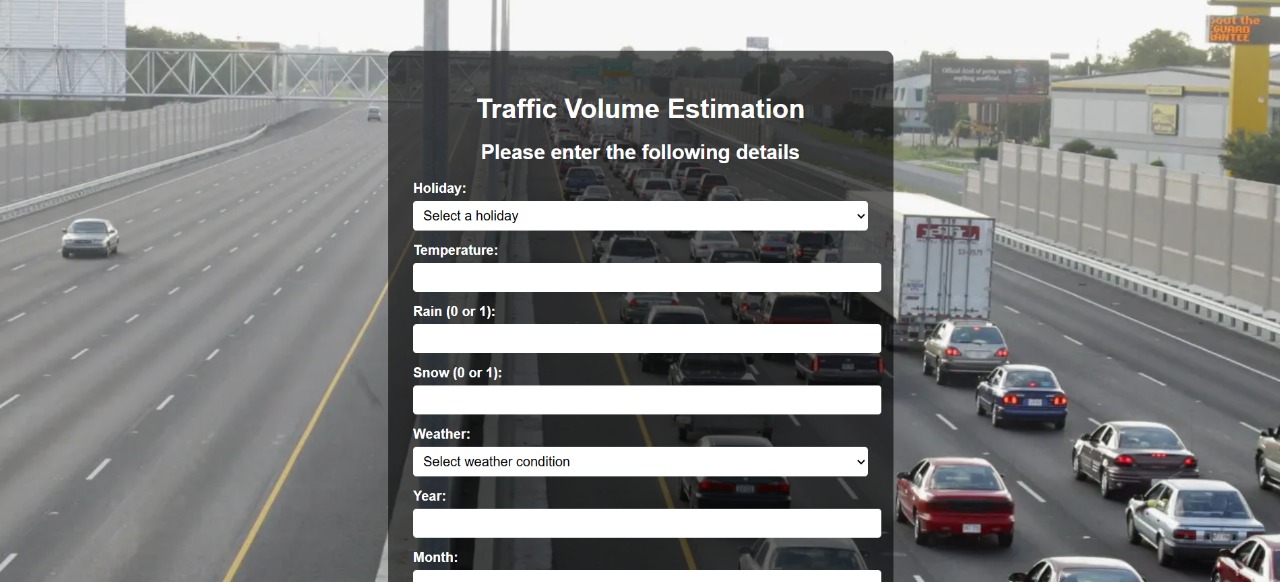
- No image or video input; purely tabular prediction  
- Does not use computer vision or YOLO (for future enhancement)  
- Accuracy depends on quality and diversity of training data  
- Model may mispredict extreme conditions or rare holidays

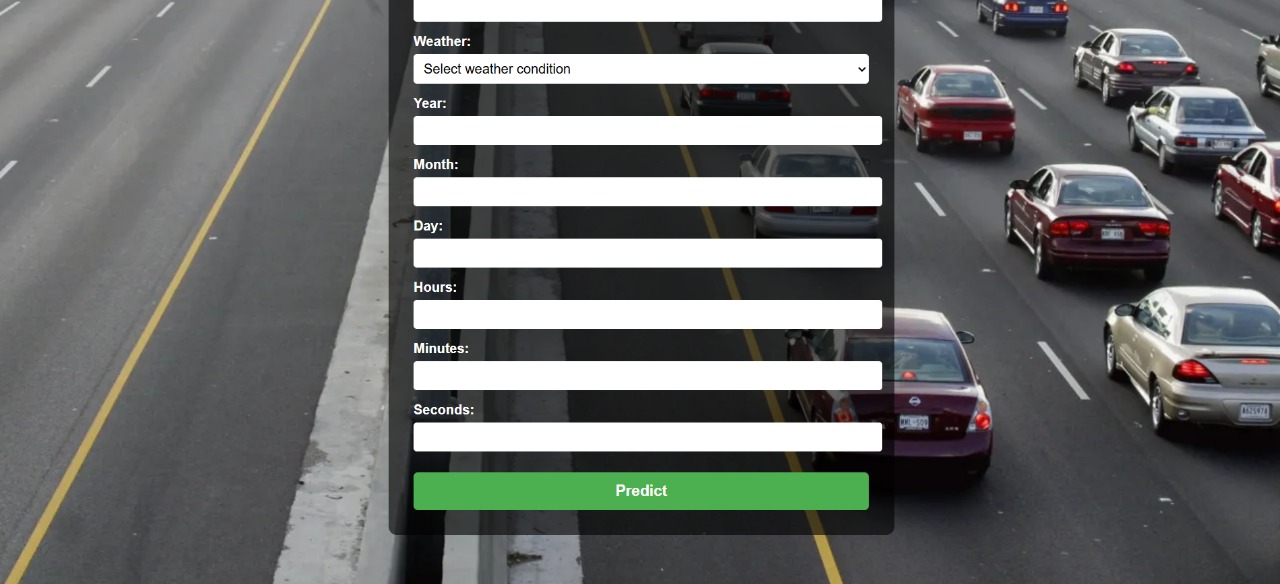
# 11. Demo and Screenshots

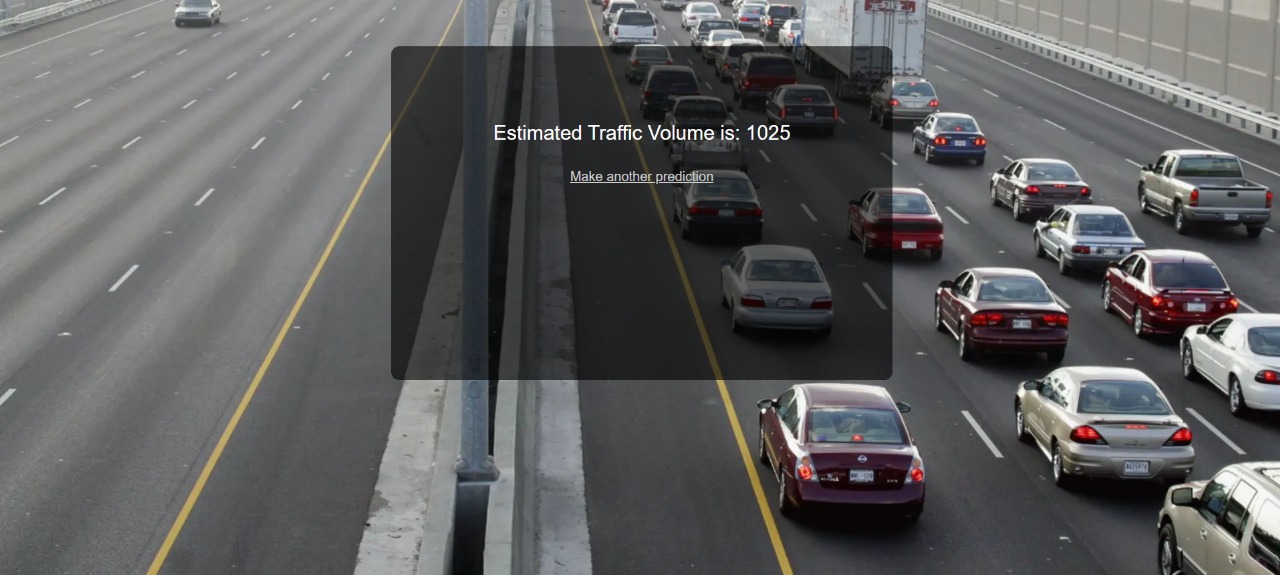
* Demo Video link :

[Link](https://drive.google.com/file/d/1SIRm1V-2SZ0hAA2pLvrItIlZ877AF455/view?usp=sharing)

* Screenshots:







# 12. Future Enhancements

* Integrate YOLO or other object detection models to enable video/image-based traffic volume estimation.
* Allow real-time camera stream input for continuous vehicle counting and analysis.
* Add database integration (e.g., MongoDB or SQLite) to store prediction history and user inputs.
* Implement visual analytics dashboards using tools like Chart.js or Plotly for data trends.
* Include location-aware predictions using GPS data or external traffic APIs.
* Build a mobile-responsive version or convert the app into a PWA (Progressive Web App).
* Optimize model performance and response time, especially for large-scale inputs.
* Add multi-user authentication with role-based access and user-specific history.
* Enable cloud deployment on platforms like Render, Replit, or Vercel with persistent storage.
* Provide complete API documentation using Swagger or Postman collections for developers.